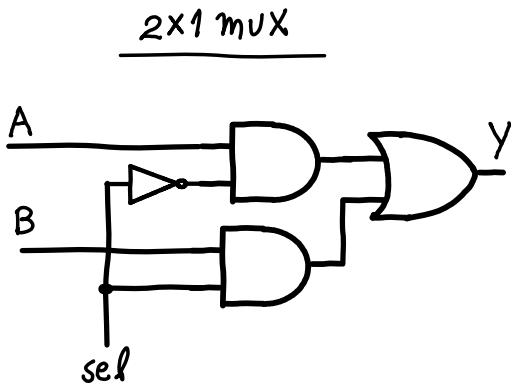
Lec-2-HW-2-2X1mux



(1.) At right is a drawing of a logic circuit. Recreate it in Electric yourself using TTL.jelib. That is, open your lib/TTL.jelib, create a new cell "mux", drop in appropriate icons from TTL.jelib, and connect wires to implement an equivalent circuit. Make A, B, sel, and Y exports (input, input, input, output). Make mux's icon view.

NB--You cannot attach a port to an icon's port, you need wire and a pin.

(2.) Make an upper-level cell "test1" and drop in a mux icon. Attach wires to its ports. Name the wires, A, B, sel, and Y. Drop in a verilog code box (^Components.Misc.VerilogCode). In the verilog box, declare Asrc, Bsrc, and SELsrc as "reg" type, and "assign" them to A, B, and sel. Then, write verilog code to effect the following time-line sequence:

A=0, B=0, sel=0
A=1
A=0
A=1
A=0
A=0, B=1, sel=0
A=1
A=0
A=1
A=0

$display the values for A, B, sel, and Y that result. What is the purpose of "sel"? What would happen if you swapped the roles of A and B, setting sel=1?

NB--Ignore warnings for the GND and Vdd ports. They are not used, but are just there to remind you that 74' chips need power and ground connections.

A side comment
Electric has a BUFF, which can be turned into a NOT:

1.) Select the BUFF's **output** pin (crosshair)
2.) extend a wire {cell area}^
3.) Select the BUFF's output pin, again
4.)
^Edit.TechnologySpecific.TogglePortNegation

Also works for NAND, NOR. **Do not** put a bubble on an **input** pin.
(Not that you need this for this exercise: you already have the 7404.)

NB--Recall that you can drive an input wire or bus by declaring a "reg" to drive it. For instance, "reg Asrc" can drive wire "A", and "Csrc" can drive bus C:

```
wire A;
wire [1:0] C;
reg Asrc;
reg [1:0] Csrc;
assign A = Asrc;
assign C = Csrc;

initial begin
   Asrc = 1;
   Csrc = 2'b01;       //---  C[1] <== 0
                       //---  C[0] <== 1
   ...
end
```

HINT: Make delays of at least #2 between assignments. Here's an "always" for $display()

```
always @( A or B or sel) begin
   #1 $display("time=%0d, A=%b, B=%b, sel=%b, Y=%b", $time, A, B, sel, Y);
end
```

(3.) There is an alternate way of looking at the same circuit. Make a new cell, test2, and drop in a mux icon. Attach wires D0 to the A port, D1 to the B port, IN to the sel port, and OUT to the Y port. Write verilog code for this sequence:

D0 = 1, D1 = 0
IN = 0
IN = 1
IN = 0
IN = 1

and $display all signals. Treating OUT as a function of IN, what function is implemented by the circuit? Repeat the above, but set D0 = 0 and D1 = 1. Now what function is implemented?

(4.) Implement your circuit on your breadboard. Check that it functions as your testbench code says it should.

What to turn in:

Turn in a cover sheet with answers to the above questions. Include a diagram of your breadboard circuit and comments on how well it conforms to the intended design (the logic circuit shown at top and your verilog circuit simulation). Use svn to check in the changes you made to TTL.jelib.